**Mohammed Abdul-Mutahhar**

**The Results – model made in python**

My code(FinishedModel.py) is attached to this file in case you want to run it yourself, (must be ran in same file as the two data sets, also requires the following modules: sklearn, panda, numpy. Matplotlib and seaborn)

**Customer IDs to invite:**

759486 417570 368669 627012 967194 537474 827098 966620 220553 89881

886190 788512 757211 341030 203339 683555 718356 310456 96282 151672

44040 952236 710856 250820 162614 983816 521331 759493 934979 537136

298491 799047 951850 679505 513772 844759 143718 233147 952834 954564

178530 759577 269347 183749 360606 422088 261484 870976 582438 458388

66600 991764 407582 257101 682039 915108 59522 428355 918989 724917

475358 718182 684062 245516 676429 295261 543816 78042 896322 809760

507909 380613 19893 975815 902402 821142 252630 878159 327091 920013

901407 581483 924073 166783 548356 856965 856226 144972 180713 967377

50062 555647 13272 308902 2455 788980 388322 665738 315298 928471

90079 101838 467990 71672 686984 878786 289116 801769 752111 792658

857924 396419 494889 327811 167200 139827 753353 321845 217497 762436

357898 441897 231982 575795 615842 116836 647688 988411 943066 743752

669508 409951 313336 729988 212450 745720 454121 90688 657186 13213

872577 554537 589591 366903 508250 680581 775261 492741 188557 720235

146011 476414 757393 104861 714512 491990 311369 192708 384921 194083

351215 539566 998585 676927 786105 578522 451936 810203 880047 367854

666849 808017 349136 505087 207870 721003 615608 421494 246961 732266

196171 238886 911311 296367 813781 793057 371883 447164 303534 887621

272987 380898 720794 319581 105668 495971 97198 46337 311119 463863

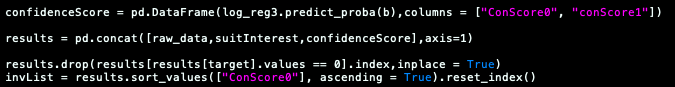
23306 472432 189054 850499 878362 249812 882146 916619 622018 163081

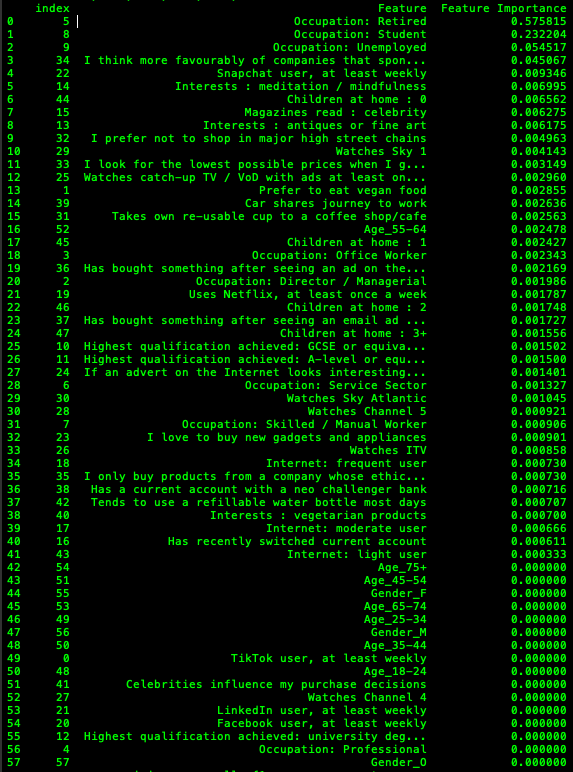
421942 977827 456240 516851 390296 921195 511229 563549 953551 865937

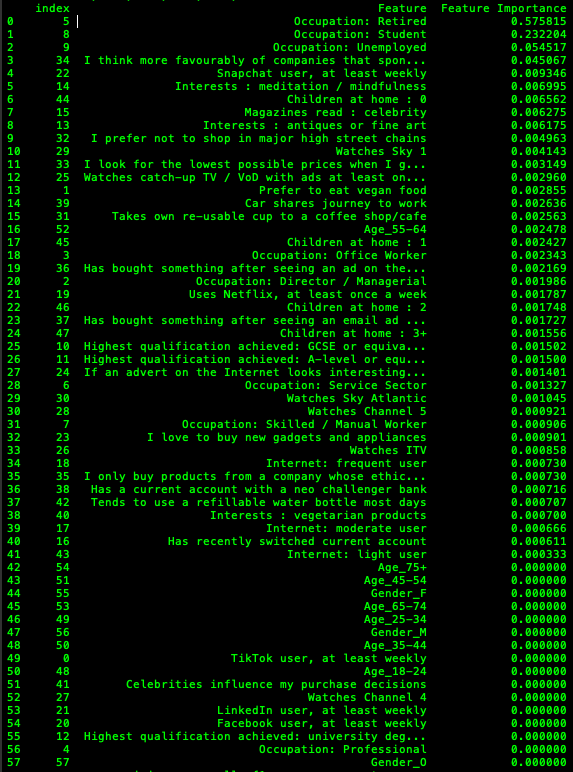
891366 178936 162784 24916 803389 96980 566677 798125 434650 895374

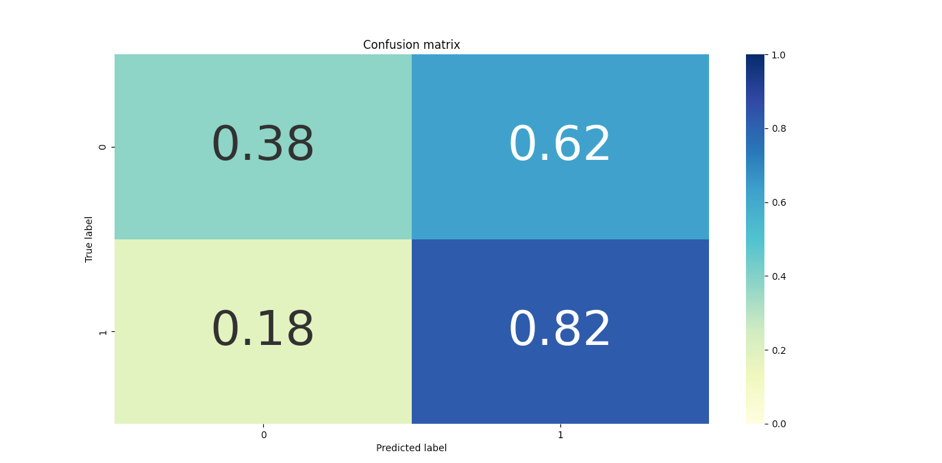
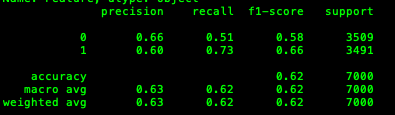
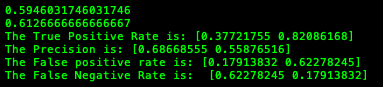
244708 233483 309945 970227 955344 68337 147583 684634 354862 844596

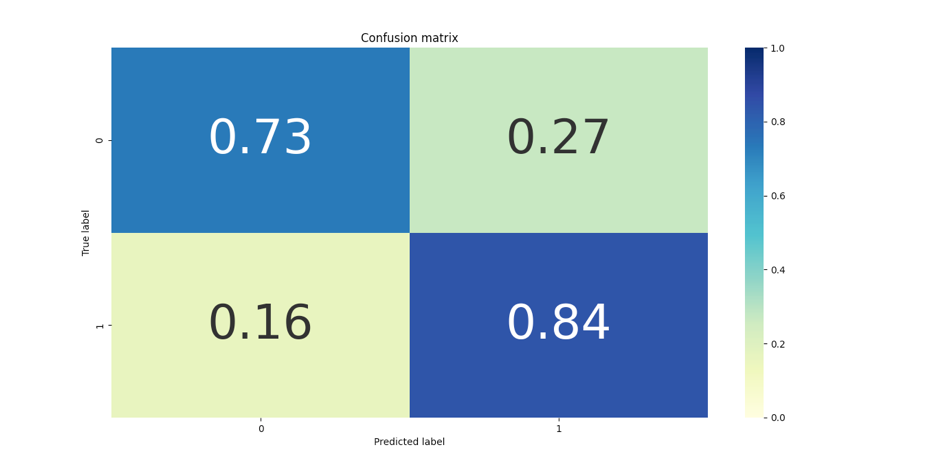
481391 181263 561433 995351 401244 387896 92530 164655 658129 40345

****These 250 customer IDs are achieved through my model, which ordered them in terms of probability score- the likelihood of each customer being interested in the sustainable range, calculated and here in my code: (I then picked the first 250)

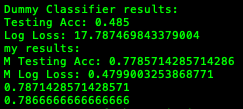
**Driving Features**

****Here you can see all of the Features ranked in order of importance to my model. All features that had an importance of 0 was ignored when creating/training my model

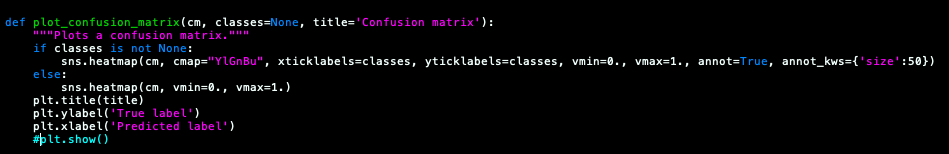
**Some Stats**

Here you can see the confusion matrix of my model after fitting it to the training set with no prior optimization aside from selecting certain features that are important (removing unimportant ones). On the right you can see some more information, regarding the precision and accuracy, the first two outputs (0.59…,0.61…) represent the score value, as calculated using the sklearn module. Log loss value for both was also 9+

Here is the confusion matrix after optimization, you can see a huge improvement in regards to the predictions of 0s as well as improvement in the prediction over all. On the right you can see more of the improvement in the testing accuracy as well as the scores(lower2) and the log loss value.

Here finally you can see my results against a dummy classifier (simple ml built into sklearn)

You can see that the dummy classifier(which calculates based on frequency of the features) is only accurate ~50% of the time were as mine is ~79%, I also have a lot less log loss. Overall, I will say that whilst my model is not close to perfect, it is much better than other simple method at a near 80% accuracy.

All of this is viewable and runnable in the code I attached however I did turn off the confusion plots, to avoid slowing down the computer, you may turn it on again by uncommenting this: